CS331- Introduction to Artificial Intelligence

Instructor: Yasir Mehmood

Assignment-1

Submission Deadline: 12 February, 2017

(There will be No Extensions)

Prepared By: Baber Khalid

Honor Code:

You are to do this assignment by yourself. All of the submitted code should be your creation and a result of your own thought process. No cooperation is allowed under any circumstances. Any help outside of the course staff is prohibited. You are also given the responsibility of reporting any cooperation incidences to the course staff. All code will be checked for similarities, and cases will be promptly forwarded to the Disciplinary Committee for action.

An Important Note:

Please submit a zip file with its name as your roll number, e.g. 1910xxxx.zip, with just the four below mentioned files in.

We will be trying to automate the checking process so please follow the guidelines in each problem to avoid any inconvenience. On top of that, it does not mean that we won't check your code to understand your logic if you have gone wrong somewhere. If your code encounters a runtime error or does not run at all then a major portion of your grade will be cut off. So **please start early on this assignment as some you might even need to revise their OOP(Object Oriented Programming Concepts).**

Problem 1: First Twenty Palindromic Primes (FileName: primePalindrome.py)

Write a simple python program which prints first twenty numbers which are palindromes as well as primes.

Prime Numbers: Those numbers which are not divisible by any other number except for 1 and themselve, e.g. 5, 7, 13.
Palindromes: Those numbers which are same when read either way, e.g. 101, 1001, 2002.

Output Format:
First Number
Second Number
Third Number

Problem 2: Make the Queue (FileName: queue.py)

We have provided you with a simple definition of stack class. Use that as reference and implement a queue class.
Please name your class and its methods as instructed at the end of this question to avoid problems, since we will be automating the checking process.

Stack: A structure which observes LIFO (Last in First Out Policy). In other words, the elements are pushed in later are emitted first.
Queue: A structure which observes FIFO(First in First out Policy) policy.

Class and Method Names:
ClassName: Queue
Method1:enqueue (returns nothing)
Method2: dequeue (returns the element removed from queue)
Method3: front (returns the next element to be removed from queue)
Method4: end (returns the last element to be removed from queue)

The purpose of next two problems is not to polish your python but your problem solving skills as you are going to need those throughout the course. So it will be for your own benefit that you solve these problems yourself and implement their solution. Also Please write the logic behind your solutions in form of comments for these problems at the top of your code. You will only get partial marks without an explanation.

Problem 3: Sheep Counting (FileName: sheep.py)

In this problem input will be given from a file named "sheep.txt". So google file reading in python if u are having problems. Output should be in a file named "sheepResult.txt".

Bleatrix Trotter the sheep has devised a strategy that helps her fall asleep faster. First, she picks a number **N**. Then she starts naming **N**, 2 × **N**, 3 × **N**, and so on. Whenever she names a number, she thinks about all of the digits in that number. She keeps track of which digits (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9) she has seen at least once so far as part of any number she has named. Once she has seen each of the ten digits at least once, she will fall asleep. Bleatrix must start with **N** and must always name (*i* + 1) × **N** directly after *i* × **N**. For example, suppose that Bleatrix picks **N** = 1692. She would count as follows:

- **N** = 1692. Now she has seen the digits 1, 2, 6, and 9.
- 2**N** = 3384. Now she has seen the digits 1, 2, 3, 4, 6, 8, and 9.
- 3**N** = 5076. Now she has seen all ten digits, and falls asleep.

What is the last number that she will name before falling asleep? If she will count forever, print `INSOMNIA` instead.

Input Format: The first line of the input gives the number of test cases, T. T test cases follow. Each consists of one line with a single integer N, the number Bleatrix has chosen.

Output Format: For each test case, output one line containing `Case #x: y`, where x is the test case number (starting from 1) and y is the last number that Bleatrix will name before falling asleep, according to the rules described in the statement.

Sample:

Input	Output
5	
0	Case #1: INSOMNIA
1	Case #2: 10
2	Case #3: 90
11	Case #4: 110
1692	Case #5: 5076

Problem 4: Happy-Blank Pancakes (pancake.py)

The input will come from a file named “pancake.txt” and output should be in a file “pancakeResult.txt”.

The Infinite House of Pancakes has just introduced a new kind of pancake! It has a happy face made of chocolate chips on one side (the "happy side"), and nothing on the other side (the "blank side").
You are the head waiter on duty, and the kitchen has just given you a stack of pancakes to serve to a customer. Like any good pancake server, you have X-ray pancake vision, and you can see whether each pancake in the stack has the happy side up or the blank side up. You think the customer will be happiest if every pancake is happy side up when you serve them.
You know the following maneuver: carefully lift up some number of pancakes (possibly all of them) from the top of the stack, flip that entire group over, and then put the group back down on top of any pancakes that you did not lift up. When flipping a group of pancakes, you flip the entire group in one motion; you do *not* individually flip each pancake. Formally: if we number the pancakes 1, 2, ..., *N* from top to bottom, you choose the top *i* pancakes to flip. Then, after the flip, the stack is *i*, *i*-1, ..., 2, 1, *i*+1, *i*+2, ..., *N*. Pancakes 1, 2, ..., *i* now have the opposite side up, whereas pancakes *i*+1, *i*+2, ..., *N* have the same side up that they had up before.
For example, let's denote the happy side as + and the blank side as -. Suppose that the stack, starting from the top, is --+-. One valid way to execute the maneuver would be to pick up the top three, flip the entire group, and put them back down on the remaining fourth pancake (which would stay where it is and remain unchanged). The new state of the stack would then be -++-. The other valid ways would be to pick up and flip the top one, the top two, or all four. It would not be valid to choose and flip the middle two or the bottom one, for example; you can only take some number off the top.
You will not serve the customer until every pancake is happy side up, but you don't want the pancakes to get cold, so you have to act fast! What is the smallest number of times you will need to execute the maneuver to get all the pancakes happy side up, if you make optimal choices?

Input Format:
The first line of the input gives the number of test cases, T. T test cases follow. Each consists of one line with a string S, each character of which is either + (which represents a pancake that is initially happy side up) or - (which represents a pancake that is initially blank side up). The string, when read left to right, represents the stack when viewed from top to bottom.
Output Format:
For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the minimum number of times you will need to execute the maneuver to get all the pancakes happy side up.

Below is a sample input and output with some explanations.

Input	Output
5	Case #1: 1
-	Case #2: 1
-+	Case #3: 2
+-	Case #4: 0
+++	Case #5: 3
--+-	

Sample Explanation:

- In Case #1, you only need to execute the maneuver once, flipping the first (and only) pancake.
- In Case #2, you only need to execute the maneuver once, flipping only the first pancake.
- In Case #3, you must execute the maneuver twice. One optimal solution is to flip only the first pancake, changing the stack to --, and then flip both pancakes, changing the stack to ++. Notice that you cannot just flip the bottom pancake individually to get a one-move solution; every time you execute the maneuver, you must select a stack starting from the top.
- In Case #4, all of the pancakes are already happy side up, so there is no need to do anything.
- In Case #5, one valid solution is to first flip the entire stack of pancakes to get +-++, then flip the top pancake to get --++, then finally flip the top two pancakes to get ++++.

About Python:

Python is not a difficult language to learn as you will see while attempting this assignment. Following are some links that will help you get started:

https://www.programiz.com/python-programming
https://www.tutorialspoint.com/python/
http://thepythonguru.com/
http://www.w3resource.com/python/python-tutorial.php/